Federated Learning in Adversarial Settings

Raouf Kerkouche, Gergely Ács, Claude Castelluccia

Federated Learning in Adversarial Settings

Raouf Kerkouche
raouf.kerkouche@inria.fr
Privatics team, Univ. Grenoble Alpes,
Inria, 38000 Grenoble, France

Gergely Ács
acs@crysys.hu
Crysys Lab, BME-HIT

Claude Castelluccia
claude.castelluccia@inria.fr
Privatics team, Univ. Grenoble Alpes,
Inria, 38000 Grenoble, France

ABSTRACT

Federated Learning enables entities to collaboratively learn a shared prediction model while keeping their training data locally. It prevents data collection and aggregation and, therefore, mitigates the associated privacy risks. However, it still remains vulnerable to various security attacks where malicious participants aim at degrading the generated model, inserting backdoors, or inferring other participants' training data. This paper presents a new federated learning scheme that provides different trade-offs between robustness, privacy, bandwidth efficiency, and model accuracy. Our scheme uses biased quantization of model updates and hence is bandwidth efficient. It is also robust against state-of-the-art backdoor as well as model degradation attacks even when a large proportion of the participant nodes are malicious. We propose a practical differentially private extension of this scheme which protects the whole dataset of participating entities. We show that this extension performs as efficiently as the non-private but robust scheme, even with stringent privacy requirements but are less robust against model degradation and backdoor attacks. This suggests a possible fundamental trade-off between Differential Privacy and robustness.

KEYWORDS

Differential Privacy, Privacy-preserving, Security, Federated learning, Robustness, Bandwidth efficient.

1 INTRODUCTION

In standard centralized training, a machine learning model is generated by a single server who collects the training data from different sources such as mobile devices, sensors, or organizations. However, data owners are often reluctant to share their potentially sensitive data with an untrusted server. To overcome this shortcoming, *Collaborative Learning* allows several parties (clients) to build a common model without sharing their private training data. It proposes to distribute and run the Machine Learning algorithms on the entities that own the data instead of a central server. Data owners periodically synchronize their local models either distributively or through a central, perhaps untrusted server. For example, in Federated Learning, clients send their model updates to the central server which then summarizes the weights into a common model and returns this model to the clients for another round. This protocol repeats until the model converges. Federated Learning has been gaining popularity and considered to train shared models for many applications such as input text prediction, ad selection[1], drug discovery[2], or various medical applications [15] over the confidential data of many different entities.

[1]https://blog.chromium.org/2019/08/potential-uses-for-privacy-sandbox.html
[2]https://www.melloddy.eu

Although Collaborative learning creates new opportunities, it also has a few drawbacks. First, it is *not robust* against misbehaving parties who may not follow the learning protocol faithfully in order to degrade the model performance. For example, a malicious party may send bogus model updates for aggregation which degrades the overall model quality or introduces backdoors [9]. Second, malicious parties can potentially *extract private information* about the training data of honest parties from their model updates or the common model [28, 31]. Third, the *bandwidth requirement* of Federated Learning can be significant for large models: each update is typically composed of $32 \cdot n$ bits, where n is the number of model parameters. Since it is not unusual to have models with thousands or even millions of parameters, the size of each update can be quite large.

The motivation of this paper is to propose a new Federated Learning scheme that mitigates these issues, and is (1) *bandwidth efficient*, (2) *privacy-preserving*, (3) *secure*, and (4) *accurate*. We identify several trade-offs between these design goals suggesting that satisfying all these requirements simultaneously is inherently difficult. More specifically, we make the following contributions:

- We adapt the signSGD learning algorithm [7, 8] to the federated learning setting which sends only a single bit per model parameter for aggregation instead of their actual value. This extreme quantization reduces the required updates' bandwidth by a factor of 32, while still providing similar performance to the standard centralized federated learning approach. As only a small random subset of participants send their updates at each federated round, our proposal is not only more bandwith efficient but also provides stronger robustness and privacy guarantees than the standard signSGD algorithm (see Section 6 for a detailed comparison).
- We propose a privacy-preserving federated signSGD extension which provides client-level Differential Privacy (DP). Specifically, it hides any information that is unique to a client's training data, regardless whether it is about a single or multiple records, but still allows learning about characteristics that are common among multiple clients' training data. We show that our DP learning protocol, whose convergence rate is also computed analytically, produces models with an accuracy comparable to the non-private federated case, even with stringent privacy guarantees (e.g., $\varepsilon = 1$).
- In order to diminish the communication costs of our DP algorithm, we propose a novel discretized and distributed version of the Gaussian Mechanism. In particular, as opposed to the standard Gaussian Mechanism [16], the noise values come from a discretized domain and are tightly concentrated around its mean depending on the desired privacy guarantee ε. As a result, these values can be encoded with fewer bits than if they came from a continuous Gaussian distribution.

Algorithm 1: StdFed: Federated Learning

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \mathbf{w}_t^k = \textbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \sum_k \frac{|D_k|}{\sum_j |D_j|} \Delta \mathbf{w}_t^k$
  end
  Output: Global model $\mathbf{w}_t$

Client$_k(\mathbf{w}_{t-1}^k)$:
  $\mathbf{w}_t^k = \text{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  Output: Model update $(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$

- We experimentally evaluate the robustness of our schemes by implementing and testing several State-of-the-Art security attacks such as model degradation (where the adversary aims at modifying the global model) or backdoor inclusion attacks (where the adversary aims at inserting hidden backdoors). We show that, due to the quantization of model updates, the non-private federated signSGD protocol, called SignFed, is more resilient to these attacks than the standard Federated Learning scheme. However, its differentially private variant turns out to be more vulnerable to the security attacks. Indeed, the attacks are inherently concealed by the noise which is introduced to guarantee Differential Privacy.

Organization: The paper is structured as follows. Section 2 details the preliminaries including the basic federated learning algorithm (StdFed) and Differential Privacy (DP). Section 3 describes SignFed which is an adaption of signSGD [8] to the federated learning setting. The performance of SignFed and StdFed are compared in Section 3.3. A DP learning protocol for client-level privacy (DP-SignFed) is presented in Section 4, whose performance are evaluated in Section 4.3. The resistance of our protocols against various security attacks is studied in Section 5. Section 6 compares our proposal with prior work, and finally Section 7 provides a summary and additional discussions about the proposed algorithms.

2 BACKGROUND

2.1 Federated Learning (StdFed)

Algorithm 2: Stochastic Gradient Descent

Input: D : training data, T_{gd} : local epochs, $\mathbf{w}$: weights
for $t = 1$ **to** T_{gd} **do**
  Select batch $\mathbb{B}$ from D randomly
  $\mathbf{w} = \mathbf{w} - \eta \nabla f(\mathbb{B}; \mathbf{w})$
end
Output: Model $\mathbf{w}$

In federated learning [26, 35], multiple parties (clients) build a common machine learning model on the union of their training data without sharing them with each other. At each round of the training, some clients retrieve the global model from the parameter server, update the global model based on their own training data, and send back their updated model to the server. The server aggregates the updated models of all clients to obtain a global model that is re-distributed to some selected parties in the next round.

In particular, a subset $\mathbb{K}$ of all N clients are randomly selected at each round to update the global model, and $C = |\mathbb{K}|/N$ denotes the fraction of selected clients. At round t, a selected client $k \in \mathbb{K}$ executes T_{gd} local gradient descent iterations on the common model $\mathbf{w}_{t-1}$ using its own training data D_k ($D = \cup_{k \in \mathbb{K}} D_k$), and obtains the updated model $\mathbf{w}_t^k$, where the number of weights is denoted by n (i.e., $|\mathbf{w}_t^k| = |\Delta \mathbf{w}_t^k| = n$ for all k and t). Each client k submits the update $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$ to the server, which then updates the common model as follows: $\mathbf{w}_t = \mathbf{w}_{t-1} + \sum_{k \in \mathbb{K}} \frac{|D_k|}{\sum_j |D_j|} \Delta \mathbf{w}_t^k$, where $|D_k|$ is known to the server for all k (a client's update is weighted with the size of its training data). The server stops training after a fixed number of rounds T_{cl}, or when the performance of the common model does not improve on a held-out data.

Note that each D_k may be generated from different distributions (i.e., Non-IID case), that is, any client's local dataset may not be representative of the population distribution [26]. This can happen, for example, when not all output classes are represented in every client's training data. The federated learning of neural networks is summarized in Alg. 1. In the sequel, each client is assumed to use the same model architecture.

The motivation of federated learning is three-fold: first, it aims to provide confidentiality of each participant's training data by sharing only model updates instead of potentially sensitive training data. Second, in order to decrease communication costs, clients can perform multiple local SGD iterations before sending their update back to the server. Third, in each round, only a few clients are required to perform local training of the common model, which not only further diminishes communications costs but also increases robustness against temporary client failures and hence makes the approach especially appealing with large number of clients.

However, several prior works have demonstrated that model updates do leak potentially sensitive information [28, 31]. Hence, simply not sharing training data *per se* is not enough to guarantee their confidentiality.

2.2 Differential Privacy

Differential privacy allows a party to privately release information about a dataset: a function of an input dataset is perturbed, so that any information which can differentiate a record from the rest of the dataset is bounded [16].

Definition 1 (Privacy loss). *Let $\mathcal{A}$ be a privacy mechanism which assigns a value Range($\mathcal{A}$) to a dataset D. The privacy loss of $\mathcal{A}$ with datasets D and D' at output $O \in$ Range($\mathcal{A}$) is a random variable $\mathcal{P}(\mathcal{A}, D, D', O) = \log \frac{\Pr[\mathcal{A}(D)=O]}{\Pr[\mathcal{A}(D')=O]}$ where the probability is taken on the randomness of $\mathcal{A}$.*

Definition 2 ((ϵ, δ)-Differential Privacy [16]). *A privacy mechanism $\mathcal{A}$ guarantees (ε, δ)-differential privacy if for any database D and D', differing on at most one record, $\Pr_{O \sim \mathcal{A}(D)}[\mathcal{P}(\mathcal{A}, D, D', O) > \varepsilon] \leq \delta$.*

Algorithm 3: SignFed: Sign Federated Learning

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{s}_t^k = \textbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \gamma \text{sign}\left(\sum_k \mathbf{s}_t^k\right)$
  end
  Output: Global model $\mathbf{w}_t$

$\textbf{Client}_k(\mathbf{w}_{t-1}^i)$:
  $\mathbf{w}_t^k = \text{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  Output: Model update $\text{sign}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$

Intuitively, this guarantees that an adversary, provided with the output of $\mathcal{A}$, can draw almost the same conclusions (up to ε with probability larger than $1-\delta$) about any record no matter if it is included in the input of $\mathcal{A}$ or not [16]. That is, for any record owner, a privacy breach is unlikely to be due to its participation in the dataset.

Moments Accountant. Differential privacy maintains composition; the privacy guarantee of the k-fold adaptive composition of $\mathcal{A}_{1:k} = \mathcal{A}_1, \ldots, \mathcal{A}_k$ can be computed using the moments accountant method [2]. In particular, it follows from Markov's inequality that $\Pr[\mathcal{P}(\mathcal{A}, D, D', O) \geq \varepsilon] \leq \mathbb{E}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]/\exp(\lambda\varepsilon)$ for any output $O \in \mathit{Range}(\mathcal{A})$ and $\lambda > 0$. This implies that $\mathcal{A}$ is (ε, δ)-DP with $\delta = \min_\lambda \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda\varepsilon)$, where $\alpha_{\mathcal{A}}(\lambda) = \max_{D,D'} \log \mathbb{E}_{O\sim\mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]$ is the log of the moment generating function of the privacy loss. The privacy guarantee of the composite mechanism $\mathcal{A}_{1:k}$ can be computed using that $\alpha_{\mathcal{A}_{1:k}}(\lambda) \leq \sum_{i=1}^k \alpha_{\mathcal{A}_i}(\lambda)$ [2].

Gaussian Mechanism. There are a few ways to achieve DP, including the Gaussian mechanism [16]. A fundamental concept of all of them is the *global sensitivity* of a function [16].

Definition 3 (Global L_p-sensitivity). *For any function $f : \mathcal{D} \to \mathbb{R}^n$, the L_p-sensitivity of f is $\Delta_p f = \max_{D,D'} ||f(D) - f(D')||_p$, for all D, D' differing in at most one record, where $||\cdot||_p$ denotes the L_p-norm.*

The Gaussian Mechanism [16] consists of adding Gaussian noise to the true output of a function. In particular, for any function $f : \mathcal{D} \to \mathbb{R}^n$, the Gaussian mechanism is defined as adding i.i.d Gaussian noise with variance $(\Delta_2 f \cdot \sigma)^2$ and zero mean to each coordinate value of $f(D)$. Recall that the pdf of the Gaussian distribution with mean μ and variance ξ^2 is

$$\text{pdf}_{\mathcal{G}(\mu,\xi)}(x) = \frac{1}{\sqrt{2\pi}\xi} \exp\left(-\frac{(x-\mu)^2}{2\xi^2}\right) \quad (1)$$

In fact, the Gaussian mechanism draws vector values from a multivariate spherical (or isotropic) Gaussian distribution which is described by random variable $\mathcal{G}(f(D), \Delta_2 f \cdot \sigma \mathbf{I}_n)$, where n is omitted if its unambiguous in the given context.

3 SIGNFED: SIGN PROTOCOL IN THE FEDERATED LEARNING SETTING

3.1 The SignFed Protocol

In the StdFed scheme, presented in Section 2.1, each selected client sends its updated model to the central server. As discussed previously, this scheme has several drawbacks in terms of bandwidth, robustness and privacy. We propose to limit these drawbacks by quantizing the model weights as in [8]. More specifically, in the new scheme, referred to as SignFed in the rest of this paper, each client sends only the sign of every coordinate value in its parameter update vector. The server takes the sign of the sum of signs per coordinate and scales down the result with a fixed constant γ (which is in the order of 10^{-3} in practice) in order to limit the contribution of each client and adjust convergence. This scaled aggregated updates are added to the global model.

More specifically, SignFed (see Alg. 3) differs from the standard federated scheme StdFed (see Alg. 1) as follows:

(1) Each client returns $\mathbf{s}_t^k = \text{sign}(\mathbf{w} - \mathbf{w}_{t-1}^k)$ instead of $(\mathbf{w} - \mathbf{w}_{t-1}^k)$, where $\text{sign} : \mathbb{R}^n \to \{-1, 1\}^n$ returns the sign of each coordinate value of the input vector if it is non-zero and a sign chosen uniformly at random otherwise.
(2) The server sums the sign vectors $\mathbf{s}_t^k$ sent by each client k and computes the sign vector of this sum as $\text{sign}\left(\sum_k \mathbf{s}_t^k\right)$. This is equivalent to take the median of all clients' signs at every position of the update vectors. Unlike in Alg. 1, the update $\mathbf{s}_t^k$ is *not* weighted with client k's data size $|D_k|$, since that would require the client to send $|D_k|$ to the server which would enable the adversary to maliciously scale up its sign vector by sending a fabricated size of its training data.

The extreme quantization performed by SignFed reduces the communication costs of federated learning by a factor of 32 (since only one bit is sent per parameter instead of 32 bits), and also, as we will demonstrate later, improves its robustness against different attacks aiming to maliciously manipulate the common model through the updates. Note also that, if the quantized update vector is sparse, other compression techniques can further improve communication efficiency [21].

3.2 Experimental Set-up

This section describes the experimental set-up that are used to evaluate the accuracy, security and privacy of our proposals in the rest of the paper. The following datasets were used: MNIST, Fashion-MNIST, IMDB, LFW and CIFAR which is augmented from 50,000 images to 500,000 (See Appendix A.9 for more details)

3.3 Performance evaluation

In this section, we compare the performance of SignFed and StdFed using the same configuration; Table 16 summarizes the different parameter values that were used for the different datasets. For SignFed, γ, the learning rate, was set to 0.001 for all datasets[3]. N, the total number of participant clients, was set to 1000. C, the the percentage of selected clients at each round, was set to 0.1. $|D_k|$

[3]We noticed experimentally that γ should be selected between range 0.001 and 0.005. And it should be increased when DP is used.

is the training data size of client k. $|\mathbb{B}|$, the batch size, was set to 50 with CIFAR dataset, 25 with IMDB, 10 for MNIST and Fashion-MNIST datasets. T_{gd}, the local gradient descent iterations per round and per client, was set to 30, 30, 5 and 50 for MNIST, Fashion-MNIST, IMDB and CIFAR, respectively. T_{cl}, the number of rounds, was set to 100 for the MNIST, Fashion-MNIST, IMDB datasets, and 400 for CIFAR. We use two optimizers: the stochastic gradient descent (SGD) [14] with a learning rate (η) set to 0.215 and the adaptive moment estimation (Adam) [20] [14] with a learning rate set to 0.001.

The global model accuracy of StdFed and SignFed on the CIFAR, MNIST, Fashion-MNIST, IMDB datasets are compared in Table 1. The results show that the accuracy performance of both schemes over the four datasets are very similar despite the severe parameter quantization.

The bandwidth consumption is calculated by measuring the average number of bits sent by a client to the server. This is computed as ($C\times$ best_round $\times$ model_size) for SignFed, and ($32\times C\times$ best_round $\times$ model_size) for StdFed, where model_size is the number of the model parameters and best_round represents the round when we get the best accuracy over T_{cl} rounds.

4 PRIVACY-PRESERVING SIGNFED

In SignFed, a participant only sends the signs of its updates, as opposed to their actual value, hence it intuitively reveals less information about the client's dataset than the original StdFed scheme. In order to experimentally validate this intuition, we implemented the inference attack described in [28] on StdFed and SignFed[4]. The results, which are not reported in this paper for lack of space, clearly validated our intuition (the attack accuracy dropped from 92% for StdFed to 50% for SignFed). Although these results are very promising and might confirm that privacy is preserved in practice, it does not provide any strong guarantees. In order to obtain theoretically private schemes, we extend SignFed with Differential Privacy. Our goal is to design differentially private schemes that are efficient in terms of accuracy and bandwidth (even for small ε values).

4.1 Privacy Model

We consider an adversary, or a set of colluding adversaries, who can access any update vector sent by the server or any clients at each round of the protocol. A plausible adversary is a participating entity, i.e. a malicious client or server, that wants to infer the training data used by other participants. The adversary is *passive* (i.e., honest-but-curious), that is, it follows the learning protocol faithfully.

Different privacy requirements can be considered depending on what information the adversary aims to infer. In general, private information can be inferred about:

- any record (user) in any dataset of any client (*record-level privacy*),
- any client/party (*client-level privacy*).

To illustrate the above requirements, suppose that several banks build a common model to predict the creditworthiness of their customers. A bank certainly does not want other banks to learn the financial status of any of their customers (record privacy) and perhaps not even the average income of all their customers (client privacy).

Record-level privacy is a standard requirement used in the privacy literature and is usually weaker than client-level privacy. Indeed, client-level privacy requires to hide any information which is unique to a client including perhaps all its training data.

We aim at developing a solution that provides *client-level privacy and is also bandwidth efficient*. For example, in the scenario of collaborating banks, we aim at protecting any information that is unique to each single bank's training data. The adversary should not be able to learn from the received model or its updates whether any client's data is involved in the federated run (up to ε and δ). We believe that this adversarial model is reasonable in many practical applications when the confidential information spans over multiple samples in the training data of a single client (e.g., the presence of a group a samples, such as people from a certain race). Differential Privacy guarantees plausible deniability not only to any groups of samples of a client but also to any client in the federated run. Therefore, any negative privacy impact on a party (or its training samples) cannot be attributed to their involvement in the protocol run.

4.2 Client-Based Privacy Preserving Federated Learning (DP-SignFed)

To guarantee differential privacy per client, every client should add enough noise to its update locally such that the server cannot learn any client-specific information from the noisy update. However, this approach (aka, local differential privacy [17]) requires so much perturbation that it is impractical if the number of clients is limited. Instead, likewise [38], we follow a different approach where clients themselves add noise in a distributed manner so that the aggregated updates are sufficiently noised to have meaningful differential privacy. To this end, individual noisy updates are encrypted with a simple and efficient encryption scheme taken from [4, 11]. The purpose of this encryption is to prevent the adversary from accessing the individual (and weakly-noised) update per client but only their sum over all clients which is in turn sufficiently noised to guarantee DP for any client.

Specifically, each client k first computes the gradient update $\Delta\mathbf{w}_t^k$ (in Line 12 of Alg. 4) and then takes the sign vector of this update. Then, a random noise share ρ_k is added to the sign vector $\mathsf{sign}(\Delta\mathbf{w}_t^k)$ so that $\sum_{k\in\mathbb{K}} \mathsf{sign}(\Delta\mathbf{w}_t^k) + \rho_k$ satisfies differential privacy. A simple solution is that $\rho_i \sim \mathcal{G}(0, \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$, which means that $\sum_{k\in\mathbb{K}} \mathsf{sign}(\Delta\mathbf{w}_t^k) + \sum_{k\in\mathbb{K}} \rho_k = \sum_{k\in\mathbb{K}} \mathsf{sign}(\Delta\mathbf{w}_t^k) + \mathcal{G}(0, \sqrt{n}\mathbf{I}\sigma)$ as the sum of Gaussian random variables also follows Gaussian distribution[5]. Indeed, the variance of the Gaussian noise has to be proportional to the L_2-sensitivity of the sign vector which is no more than $\sqrt{n}$, where n is the number of parameters.

However, recall that the adversary can access $\mathsf{sign}(\Delta\mathbf{w}_t^k) + \rho_k$, which means that, if $|\mathbb{K}|$ is too large, ρ_k is likely to be small allowing the adversary to learn $\mathsf{sign}(\Delta\mathbf{w}_t^k) + \rho_k$ very accurately. For this reason, each client k encrypts $\mathsf{sign}(\Delta\mathbf{w}_t^k) + \rho_k$ and sends the encrypted result to the aggregator. After summing all the encrypted values,

[4] A model is trained for gender classification on the LFW dataset. The adversary's goal is to infer from the model updates whether a specific group of individuals in a client's dataset are black.

[5] More precisely, $\sum_i \mathcal{G}(v_i, \xi_i) = \mathcal{G}\left(\sum_i v_i, \sqrt{\sum_i \xi_i^2}\right)$

Table 1: Model accuracy and average bandwidth consumption from a client to the server (Megabytes) based on the best round (over T_{cl} rounds) in terms of accuracy.

Dataset	StdFed			SignFed		
	Acc	round	Cost	Acc	round	Cost
CIFAR	0.86	375	205.46	0.83	386	6.61
MNIST	0.99	88	58.55	0.98	48	1.0
Fashion-MNIST	0.89	90	59.88	0.87	68	1.41
IMDB	0.88	84	13.53	0.85	91	0.46

the server obtains $\sum_k \text{Enc}_{\mathbf{K}_k}(\text{sign}(\Delta\mathbf{w}_t^k) + \rho_k) = \sum_k \text{sign}(\Delta\mathbf{w}_t^k) + \mathcal{G}\left(0, \sqrt{n}\sigma\mathbf{I}\right)$ where $\text{Enc}_{\mathbf{K}_k}(\text{sign}(\Delta\mathbf{w}_t^k) + \rho_k) = \text{sign}(\Delta\mathbf{w}_t^k) + \rho_k + \mathbf{K}_k \mod m$ and $\sum_k \mathbf{K}_k = \mathbf{0}$ (see [4, 11] for details). Here, modulo is taken element-wise and $m = 2^{\lceil \log_2(\max_k ||1+\rho_k||_\infty |\mathbb{K}|)\rceil}$. Therefore, the server can only access the aggregate which is sufficiently noised to guarantee differential privacy; any client-specific information that could be learnt from the noisy aggregate is quantified by the moments accountant described in Section 2.2. To make learning more resilient to perturbation, the server takes the sign of the sum of updates and scales the result with $\gamma < 1$ which is crucial to achieve convergence in practice especially if $\sqrt{n}\sigma$ is large.

Unfortunately, the above simple approach is not bandwidth efficient; adding noise from the continuous domain requires each noisy update $\text{sign}(\Delta\mathbf{w}_t^k) + \rho_k$ to be encoded as a floating-point number[6] (represented by at least 32 bits on a commodity hardware) no matter that $\text{sign}(\Delta\mathbf{w}_t^k)$ would need only 1 bit per coordinate. Therefore, the noisy update needs at least 32 times more data to be transferred from a client to the server than with SignFed (in Alg. 3).

To alleviate the above bandwidth problem, each client k generates a random integer from a *discrete* Gaussian distribution with mean $\text{sign}(\Delta\mathbf{w}_t^k)$, encrypts this random integer, and sends the result for aggregation. Since the discrete Gaussian random variable has an integer value and is concentrated around its mean, its value can be encoded with fewer bits than a floating-point number. The new learning algorithm, called DP-SignFed, guarantees differential privacy for any client and is summarized in Alg. 4.

In what follows, we first describe the Discrete Gaussian Mechanism (DGM), which is used in DP-SignFed, and prove that it practically provides the same privacy guarantee as the continuous Gaussian Mechanism (GM) if its variance is sufficiently large. This allows us to precisely quantify the privacy guarantee of DP-SignFed. Finally, we show that using DGM instead of (continuous) GM in DP-SignFed reduces the communication overhead by roughly 40%.

4.2.1 Discrete Gaussian Mechanism (DGM). The discrete Gaussian distribution has probability mass function

$$\text{pmf}_{\mathcal{DG}(\mu,\xi)}(x) = Z^{-1}\exp(-(x-\mu)^2/2\xi^2) \tag{2}$$

where $Z = \sum_{x\in\mathbb{Z}}\exp(-(x-\mu)^2/2\xi^2)$. Note that $\mu \in \mathbb{R}$ but the support of $\mathcal{DG}$ is always $\mathbb{Z}$. Although Z is infeasible to compute, there are several efficient techniques to sample from the discrete Gaussian distribution [29].

The next lemma shows that the pmf of the discrete Gaussian distribution can be almost perfectly approximated by its continuous counterpart if ξ is large enough.

[6] and then as a large integer for encryption

Lemma 1. *Let* $\text{pmf}_{\mathcal{DG}(\mu,\xi)}(x)$ *and* $\text{pdf}_{\mathcal{G}(\mu,\xi)}(x)$ *be as defined in Eq. (2) and Eq. (1), respectively, and* $\kappa(\xi) = \frac{2e^{-2\pi^2\xi^2}}{1-e^{-6\pi^2\xi^2}}$. *Then,* $1 - \kappa(\xi) \le \frac{\text{pdf}_{\mathcal{G}(\mu,\xi)}(x)}{\text{pmf}_{\mathcal{DG}(\mu,\xi)}(x)} \le 1 + \kappa(\xi)$ *for* $x \in \mathbb{Z}$.

The proof can be found in Appendix A.1.

The multivariate spherical version of $\mathcal{DG}$ can be defined analogously to the spherical Gaussian distribution, that is, if $\mathbf{z} \sim \mathcal{DG}(\boldsymbol{\mu}, \xi)$, then $z_i \sim \mathcal{DG}(\mu_i, \xi)$ independently for each i.

The Discrete Gaussian Mechanism (DGM) is defined analogously to the (continuous) Gaussian Mechanism except that it uses discrete Gaussian noise instead of its continuous counterpart for perturbation. The next theorem shows that the moments of DGM can be tightly upper bounded by that of the continuous Gaussian mechanism if ξ is large enough, and hence the privacy guarantee of DGM can be efficiently and accurately approximated.

Let $\eta_0^{\mathcal{G}}(x|\xi) = \text{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1^{\mathcal{G}}(x|\xi) = (1-C)\text{pdf}_{\mathcal{G}(0,\xi)}(x) + C\text{pdf}_{\mathcal{G}(1,\xi)}(x)$ where C is the sampling probability of a single client in a single round. Let

$$\alpha_{\mathcal{G}}(\lambda|C) = \log\max(E_1(\lambda,\xi,C), E_2(\lambda,\xi,C)) \tag{3}$$

where $E_1(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_0^{\mathcal{G}}(x|\xi,C) \cdot \left(\frac{\eta_0^{\mathcal{G}}(x|\xi,C)}{\eta_1^{\mathcal{G}}(x|\xi,C)}\right)^\lambda dx$ and $E_2(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_1^{\mathcal{G}}(x|\xi,C) \cdot \left(\frac{\eta_1^{\mathcal{G}}(x|\xi,C)}{\eta_0^{\mathcal{G}}(x|\xi,C)}\right)^\lambda dx$. $\alpha_{\mathcal{DG}}(\lambda|C)$ is defined analogously to $\alpha_{\mathcal{G}}(\lambda|C)$.

Theorem 1 (Privacy of DGM). $\alpha_{\mathcal{DG}}(\lambda|C) \le \alpha_{\mathcal{G}}(\lambda|C) + \log\left(\frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}\right)$ *for any* C, *where* $\kappa(\xi)$ *is defined in Lemma 1. Therefore, DGM is* $(\min_\lambda\left(\alpha_{\mathcal{G}}(\lambda|C) + \log\left(\frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}\right)\right) - \log\delta)/\lambda, \delta)$*-DP.*

The proof can be found in Appendix A.2. Given a fixed value of δ, ε is computed numerically as in [2, 30].

4.2.2 Privacy of DP-SignFed. As shown in Alg. 4, each client k generates a random integer vector $\mathbf{z}_k \sim \mathcal{DG}(\text{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$ in DP-SignFed. Then, every client sends the encrypted result $\text{Enc}_{\mathbf{K}_k}(\mathbf{z}_k)$ to the aggregator. After summing all the encrypted integers, the server obtains

$$\sum_k \text{Enc}_{\mathbf{K}_k}(\mathbf{z}_k) = \sum_k \mathbf{z}_k = \sum_k \mathcal{DG}(\text{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|}) \tag{4}$$

The next theorem, proved in Appendix A.3, shows that DP-SignFed is differentially private, supposing that the adversary can only access $\sum_k \mathcal{DG}(\text{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$ except any of its members $\mathcal{DG}(\text{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$.

Algorithm 4: DP-SignFed: Federated Learning with Client Privacy

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients randomly
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \mathbf{w}_t^k = \textbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \gamma \cdot \text{sign}\left(\sum_k \Delta \mathbf{w}_t^k\right)$
  end
Client$_k(\mathbf{w})$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}$
  $\Delta \mathbf{w}_t^k = \text{SGD}(D_k, \mathbf{w}_t^{k-1}, T_{\text{gd}}) - \mathbf{w}_{t-1}^k$
  Output: $\text{Enc}_{K_k}\left(\mathcal{DG}\left(\text{sign}\left(\Delta \mathbf{w}_t^k\right), \sqrt{n}\mathbf{I}\sigma/\sqrt{|K|}\right)\right)$

Theorem 2 (Privacy of DP-SignFed). *For any $\delta > 0$, DP-SignFed is $(\min_\lambda(T \cdot \left(\alpha_{\mathcal{G}}(\lambda|C) + \log\left(\frac{(1+\kappa(\sqrt{n}\sigma))^\lambda}{(1-\kappa(\sqrt{n}\sigma))^{\lambda+1}}\left(\frac{1+\nu}{1-\nu}\right)^3\right)\right) - \log\delta)/\lambda, \delta)$-DP, where $\sigma \geq \sqrt{|\mathbb{K}|\ln(2+2/\nu)/2n\pi^2}$ and κ is defined in Lemma 1.*

Again, given a fixed value of δ, ε is computed numerically as in [2, 30].

4.2.3 Communication overhead. The domain of $\mathbf{z}$ in Eq. (4) is the support of $\mathcal{DG}$ which is still unbounded. This means that the size of the encrypted text can be very large though with exponentially small probability. Indeed, $||\mathbf{z}_k||_\infty$ is unbounded and hence modulo $m = 2^{\lceil \log_2(\max_k ||\mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$ has to be large. To overcome this problem, we choose modulo m to be so large that the probability that $2^{\lceil \log_2(\max_k ||\mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$ is larger than m is negligible. For this purpose, we rely on the following concentration inequality of the discrete Gaussian distribution.

Lemma 2 ([29], Lemma 2.2). *For any $\nu > 0, \xi > \sqrt{\ln(2+2/\nu)/2\pi^2}$, and $t > 0$, $\Pr_{x\sim\mathcal{DG}(\mu,\xi)}[|x-\mu| \geq t \cdot \xi] \leq 2e^{-t^2/2} \cdot \frac{1+\nu}{1-\nu}$.*

Lemma 2 implies that if $\xi = \sqrt{n}\sigma/\sqrt{|\mathbb{K}|} > 3.51$ then $\frac{1+\nu}{1-\nu} < \frac{3}{2}$ and $\Pr_{\mathbf{z}\sim\mathcal{DG}(\boldsymbol{\mu}, \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})}[||\mathbf{z} - \boldsymbol{\mu}||_\infty \geq t\sqrt{n}\sigma] \leq 3ne^{-|\mathbb{K}|t^2/2}$ after applying the union bound. For example, if $m = 2^{\lceil \log_2(12\sqrt{n}\sigma|\mathbb{K}|)\rceil}$ (i.e., $t = 12$) then the probability that $||\mathbf{z}_k - \boldsymbol{\mu}_k||_\infty$ cannot be bounded by $12\sqrt{n}\sigma$ per client is less than 2^{-80} even if $|\mathbb{K}| = 1$ and $n = 10^7$. Thus, a client needs to transfer $n \cdot \log_2\left(2^{\lceil \log_2((12\sqrt{n}\sigma + \max_k ||\boldsymbol{\mu}_k||_\infty)|\mathbb{K}|)\rceil}\right)$ bits in total to the aggregator. For example, if $|\mathbb{K}| = 100$, $\max_k ||\boldsymbol{\mu}_k||_\infty = 1$, $\sigma = 1$ (i.e., $\varepsilon \approx 0.2$), then $\log_2 m = 22$. By contrast, if noise was generated from the continuous domain, then $\log_2 m = 32$ which means that DGM reduces the communication overhead by roughly 32%.

Notice that if ε or δ is smaller (i.e., there is stronger privacy guarantee), then σ is larger which implies that m also increases, and hence more bits need to be transferred to the server per parameter. This results in a trade-off between Differential Privacy and bandwidth efficiency.

4.3 Performance evaluation

The performance of DP-SignFed is compared with DP-StdFed in Table 2 and 3. DP-StdFed is an extension of StdFed to provide client-level differential privacy. Specifically, in DP-StdFed, the randomly selected clients first clip their model update vector to have a bounded L_2-norm[7], add continuous Gaussian noise to the clipped update vector, and then transfer the *non-quantized* noisy model update to the server (see Alg. 5 in the Appendix of [3] for more details). The configurations of these protocols are summarized in Table 17 of [3].

Table 2 and 3 show the best model accuracy observed over 200 rounds with each algorithm on the MNIST and Fashion-MNIST datasets, respectively. DP-StdFed provides the best accuracy; for MNIST, it is 86-93%, and for Fashion-MNIST, it is 63-78% depending on the privacy parameter ε. The performance degradation of DP-SignFed compared to DP-StdFed is 0.02 on MNIST and 0-0.07 on Fashion-MNIST. As expected, weaker privacy requirement (i.e., larger ε) needs smaller noise magnitude and hence better accuracy for all algorithms.

The communication cost of DP-SignFed is 66% of that of DP-StdFed. Specifically, while DP-StdFed needs 32 bits per parameter, DP-SignFed requires 21-22 bits depending on the value of ε[8]. If ε is smaller, the variance of the noise is larger, and hence more bits are necessary to encode the noisy signs. Notice that even if we use early stopping and record the best accuracy sooner than 200 rounds, it will neither decrease the communication cost nor increase the privacy guarantee. Indeed, one has to execute all the 200 rounds in the first place to identify the best performing round.

Table 2: Model accuracy and communication cost on MNIST dataset. We give the communication cost per parameter value (bits/parameter) for any value of ε.

	$\varepsilon = 1$		$\varepsilon = 2$		$\varepsilon = 4$	
	Acc	Cost	Acc	Cost	Acc	Cost
DP-StdFed	0.86	32	0.92	32	0.93	32
DP-SignFed	0.87	22	0.90	21	0.91	21

Table 3: Model accuracy and communication cost with Fashion-MNIST dataset. We give the communication cost per parameter value (bits/parameter) for any value of ε.

	$\varepsilon = 1$		$\varepsilon = 2$		$\varepsilon = 4$	
	Acc	Cost	Acc	Cost	Acc	Cost
DP-StdFed	0.63	32	0.74	32	0.78	32
DP-SignFed	0.63	22	0.70	21	0.73	21

[7] The sensitivity $S = \sqrt{n}$ and $\gamma = 0.005$ for DP-SignFed. For DP-StdFed, the server computes the median L_2-norm value over N L_2-norm values received during an additional initialization round. Hence, S is set to 1.73 and 2.15 for MNIST and Fashion-MNIST, respectively.

[8] It is computed from $\log_2\left(2^{\lceil \log_2((12\sqrt{n}\sigma + \max_k ||\boldsymbol{\mu}_k||_\infty)|\mathbb{K}|)\rceil}\right)$ where σ is obtained from ε and $\delta = 10^{-5}$ using the moments accountant. This ensures that the magnitude of the noisy update per model parameter is less than the modulus n with probability at most 2^{-80} (see Section 4.2.3).

5 SECURITY ANALYSIS

This section evaluates the robustness of SignFed, DP-SignFed and DP-StdFed against several state-of-the-art security attacks.

5.1 Security Model

Adversarial model: In this work, we assume that the adversary controls a certain fraction of the participating entities/clients at each round of the training, which means it can access and modify these clients' training data as well as all parameters of their local model. We, however, assume that *the server is honest* (i.e., it does not manipulate the aggregate or the update vector sent by any client). The set of all malicious nodes is denoted by $\mathbb{M}$.

We consider two types of adversary. The first one aims at degrading the overall model performance (i.e., increase the average misclassification rate). The second one aims at causing targeted misclassification on some particular classes of samples by injecting backdoors into the model during the training phase. These adversaries are *active* in the sense that they may not follow the learning protocol faithfully.

Next, we detail the attacks considered in our work.

5.1.1 *Overall Model Degradation Attacks.*

Random Update Attack. In this attack, malicious clients, whose numbers might vary as shown later, use random updates. More specifically, instead of the true model update $\Delta\mathbf{w}_t^k$, each malicious client k generates a random update $\Delta\hat{\mathbf{w}}_t^k$ in all time slots t [10], where $\Delta\hat{\mathbf{w}}_t^k$ is drawn from an isotropic Gaussian distribution $\mathcal{G}(0, \sigma_{\text{Adv}}\mathbf{I})$ with mean zero and variance σ_{Adv}^2. Each malicious party selects the noise independently (i.e., they do not collude).

Gradient Ascent Attack. In this attack, malicious clients aim at maximizing the loss by performing gradient *ascent* instead of descent on their own training data. In particular, *every* malicious client $k \in \mathbb{M}$ updates the model parameters locally as $\mathbf{w}_t^k = \mathbf{w}_{t-1}^k + \eta_{\text{Adv}}\nabla f(\cup_{k\in\mathbb{M}} D_k; \mathbf{w})$, where η_{Adv} is set in order to suppress the updates of honest clients and to maximize the impact of their own update on the common model. Notice that this attack assumes *colluding malicious clients* (i.e., every malicious client sends exactly the same update computed on the union of their training data). This attack attempts to maximize the *average* misclassification rate of the common model, and is more effective if the number of malicious parties is large, or the training data of the malicious and benign nodes come from similar distributions.

We note that Gradient Ascent Attack is equivalent to the Sign Inversion Attack for SignFed, described in [8], if $T_{\text{gd}} = 1$ (i.e., each client computes its update using a single mini-batch in every round). In Sign Inversion Attack, all malicious clients faithfully compute the sign of their model update, but then send the *inverted* signs to the server for aggregation.

5.1.2 *Backdoor Attacks (Targeted Attacks).*

The goal of these attacks is to selectively degrade the accuracy of the common model with respect to only a few tasks. As opposed to the overall model degradation attacks, they generate *targeted* misclassification while preserving the model convergence as well as a high average prediction accuracy except, of course, for the targeted tasks, called backdoor classes.

We distinguish two types of backdoors: In-backdoors and Out-backdoors.

- *In-backdoor Attacks:* In-backdoor attacks [9] are created for a class of samples that exists in the training data of some parties. Specifically, for some training samples $D_{aux} \subseteq D_k$, each adversary uses output labels that are different from their true labels. Let y' denote the adversarially chosen label for a training sample $(x, y) \in D_{aux}$, and D'_{aux} denotes the set of all relabelled samples (i.e., $(x, y') \in D'_{aux}$). The new objective is to minimize the loss $f((D_k \setminus D_{aux}) \cup D'_{aux}; \mathbf{w})$.
- *Out-backdoor Attacks:* As opposed to in-backdoors, *out-backdoors* are created from samples that do *not* exist in the training data of any honest clients and are relabelled to have a class that *does* exist in their training data. Specifically, let L denote the set of labels that exist in $D = \cup_k D_k$. The adversary creates D''_{aux} such that, for each $(x, y) \in D''_{aux}$, $x \notin D$ and $y \in L$. The new objective is to minimize the loss $f(D_k \cup D''_{aux}; \mathbf{w})$.

To illustrate the difference between in- and out-backdoors, consider a model which recognizes dogs and rabbits in the input photos. If the adversary relabels all photos of dogs as 'rabbit' in its training data, then it is an in-backdoor attack. However, if the adversary adds new photos of frogs to its training data and relabels them as 'dog', then this is an out-backdoor attack.

Out-backdoors are more difficult to detect than in-backdoors as they can come from a much larger set of samples, which are potentially unknown to the protocol participants. Hence, out-backdoors are especially severe in security-related applications such as in access control.

As per [9], the adversary also uses explicit boosting to outbalance the combined effect of benign model updates. For both in- and out-backdoors, the adversary boosts $\Delta\mathbf{w}_{\text{Adv}}^t$ at time t by sending $\eta_{\text{Adv}}\Delta\mathbf{w}_{\text{Adv}}^t$ ($\mu > 1$) in order to suppress the model updates of benign parties. Importantly, η_{Adv} should be large enough in order to achieve misclassification of the backdoor class but also small enough to ensure the convergence of the common model and hence hide the attack.

5.2 SignFed Security Analysis

In this section, we evaluate the robustness of SignFed against the security attacks presented previously.

For the Overall Model Degradation attacks, different percentage of malicious nodes are considered, the MNIST and IMDB datasets were used, and the same experimental setting as defined in Table 16 is used. The boosting parameter η_{Adv} of the Gradient Ascent Attack is set to 10 with MNIST dataset and 20 with the IMDB dataset (we use the boosting only with StdFed). We also do not need to use boosting for the Random Update Attack with StdFed as $\sigma_{\text{Adv}} = 200$ generates large noise which prevents the model convergence.

For the Backdoor attacks, the MNIST and CIFAR datasets were used and the experimental setting is shown in Table 18. As backdoor attacks, which aim at modifying the prediction of one particular label while maintaining the global accuracy, are more difficult to perform on binary classifiers, we switched to the CIFAR dataset with a multiclass classifier. Furthermore, similarly to [9], we reduce the total number of clients N from 1000 to 10, we use different percentages of malicious nodes: 10%, 20% and 40%, and all clients

report their updates to the server at each round (i.e. $C = 1.0$). The malicious nodes collude by sharing their data for the training and by sending the same update to the server.

5.2.1 *Overall Model Degradation Attacks.*

Random update. Table 4 and 5 depict the best accuracy of the global model over 100 rounds according to the fraction of malicious nodes in set $\mathbb{K}$. The results show that SignFed is robust against the random update attack even if 20% of all nodes are malicious, while StdFed fails to converge even if 1% of all nodes are malicious. Indeed, with 20% of malicious nodes, SignFed reaches an accuracy of 98% and 86% for the MNIST and IMDB datasets, respectively. On the contrary, StdFed fails to converge even with one malicious node at each round. In fact, as we show in Appendix A.4, SignFed's convergence rate is $O\left(\frac{1}{(1-\alpha)\sqrt{CNT}_{\text{cl}}}\right)$, where α denotes the fraction of malicious clients. This is in contrast to the sign inversion attack detailed in [8], which has a convergence rate of $O\left(\frac{1}{(1-2\alpha)\sqrt{CNT}_{\text{cl}}}\right)$, that is, convergence is only possible if less than half of the nodes are malicious.

Table 4: Random update attack on SignFed and StdFed with the MNIST dataset depending on the fraction of malicious nodes. $\sigma_{\text{Adv}} = 200$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
StdFed	-	-	-	-
SignFed	0.98	0.98	0.94	-

Table 5: Random update attack on SignFed and StdFed with the IMDB dataset depending on the fraction of malicious nodes. $\sigma_{\text{Adv}} = 200$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
StdFed	-	-	-	-
SignFed	0.86	0.86	0.54	-

Gradient Ascent Attack. Table 6 and 7 show the best accuracy of the global model over 100 rounds when the adversary aims to degrade the average model performance by performing gradient ascent on its own training data. With the MNIST dataset (in Table 6), SignFed reaches an accuracy of 98% and 79% for 20% and 40% of malicious nodes, respectively, while StdFed does not converge even if only 10% of the nodes are malicious. For IMDB dataset, SignFed reaches an accuracy of 86% and 72% for 10% and 20% of malicious nodes, respectively, while StdFed fails to converge with only 10% of malicious nodes. Indeed, StdFed does not converge even if we have only one malicious node. The reason for this difference is that malicious nodes can scale up their update with η_{Adv} and hence boost its effect on the global model. However, such adversarial boosting does not work with SignFed as the trusted server accepts only the values -1 and $+1$ in the update vectors. Therefore, a single malicious client does not have larger impact on the global model than any other honest client. To boost its impact, the adversary can only increase the number of the malicious clients, as shown by the experimental results. Since Gradient Ascent is equivalent to Sign Inversion Attack if $T_{\text{gd}} = 1$, the convergence rate of Gradient Ascent in this restricted scenario is $O\left(\frac{1}{(1-2\alpha)\sqrt{CNT}_{\text{cl}}}\right)$ as shown in [8].

Table 6: Gradient ascent attack on SignFed and StdFed with the MNIST dataset. $\eta_{\text{Adv}} = 10$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
StdFed	-	-	-	-
SignFed	0.98	0.98	0.79	-

Table 7: Gradient ascent attack on SignFed and StdFed with the IMDB dataset. $\eta_{\text{Adv}} = 20$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
StdFed	-	-	-	-
SignFed	0.86	0.72	0.52	-

5.2.2 *Backdoor attacks.*

In-backdoor attack. Figure 1, 2 and Table 8, 9 show the effect of in-backdoor attacks on the MNIST and CIFAR datasets, respectively. In all experiments, there are ten clients, out of which different fraction of malicious nodes are considered. Figure 1 depicts the accuracy of the global model for MNIST, when the adversary relabels every image of digit '5' to '7' in its local dataset. The red plots show the accuracy of the global models, while the green ones display the model accuracy only for the images with label '5' (i.e., accuracy on the backdoor class). The results show that SignFed is robust as both global model accuracy and model accuracy on the specific in-backdoor class (digit 5) reach 99% by the end of the training.

By contrast, with StdFed, while the accuracy of the global model converges slowly to 99%, the accuracy of the attacked model oscillates. Similar behaviour can be observed in Figure 2 which plots the accuracy on CIFAR dataset, where images of airplanes are relabelled to 'ship' in the adversary's training data. In these experiments, StdFed fails to converge on the backdoor class, and its accuracy on CIFAR never exceeds 55%.

The oscillation of accuracy with StdFed can be explained by the nature of gradient descent and in particular backpropagation: when the malicious client injects the backdoor, it scales its update with η_{Adv}. In the following round, honest clients scale up their gradients on the backdoor samples (i.e., images of digit 5 in MNIST and images of airplanes in CIFAR) in order to "fix" the classification error on the backdoor class. In the next round, when the model is "fixed" (i.e., digit '5' is correctly predicted as '5' again), the adversary's gradients

are increased again in order to re-inject the backdoor. This process repeats till the end of the training. By contrast, and similarly to the overall model degradation attacks, a malicious client cannot scale up its update in SignFed as the update vectors must take value from $\{-1, 1\}^n$.

Table 8 shows the accuracy of the model on digit class 5 (in-backdoor class) when we consider different percentage of malicious nodes (values are chosen based on the best model accuracy over 40 rounds). The global accuracy of the model over all the classes is 99% and the accuracy on class '5' is 99% independently of the number of malicious nodes and regardless whether StdFed or SignFed is used.

As in the previous table, Table 9 shows the accuracy of the model on airplane class (in-backdoor class) when we consider different number of malicious nodes (values are chosen based on the best global accuracy over 100 rounds). SignFed with 20% of malicious nodes reaches a global accuracy of 84%, and an accuracy of 76% on the in-backdoor class. However, StdFed with the same amount of malicious nodes reaches 80% of global accuracy and 0% for the airplane class. The results confirm the larger robustness of SignFed over StdFed.

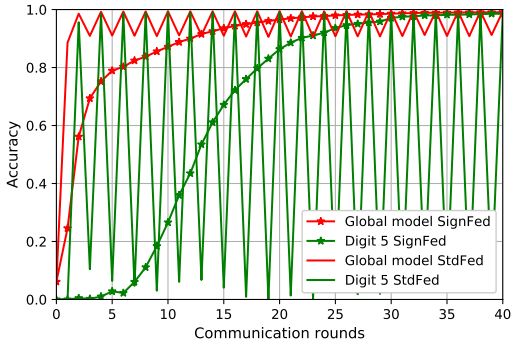

Figure 1: In-backdoor attack on SignFed and StdFed with the MNIST dataset, $\eta_{\text{Adv}} = 7$. The figure displays the global accuracy convergence and the accuracy of the label "5" which is under attack. 10% of the nodes are malicious.

Table 8: In-backdoor attack on SignFed and StdFed with the MNIST dataset, η_{Adv} is set to 7, 3, 1 for 10%, 20%, 40% respectively. The table depicts the global accuracy convergence and the accuracy of the label "5" which is under attack.

		10%	20%	40%
SignFed	Model accuracy	0.99	0.99	0.99
	Accuracy on digit class '5'	0.99	0.99	0.99
StdFed	Model accuracy	0.99	0.99	0.99
	Accuracy on digit class '5'	0.99	0.99	0.99

Out-backdoor attack. The main goal of the out-backdoor attack is to introduce fake information during the training by relabeling a sample, whose true label is not a valid output of the global model. We experimented this attack on MNIST by first excluding all samples with digit '0' in all clients' training datasets. We then choose

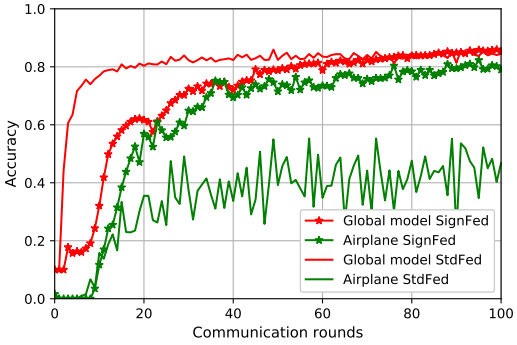

Figure 2: In-backdoor attack on SignFed and StdFed with CIFAR dataset, $\eta_{\text{Adv}} = 7$. The figure displays the global accuracy convergence and the accuracy of the label "airplane" which is under attack. 10% of the nodes are malicious.

Table 9: In-backdoor attack on SignFed and StdFed with the CIFAR dataset, η_{Adv} is set to 7, 4, 2 for 10%, 20%, 40% respectively. The table depicts the global accuracy convergence and the accuracy of the label "airplane" which is under attack.

		10%	20%	40%
SignFed	Model accuracy	0.86	0.84	0.82
	Accuracy on airplane class	0.80	0.76	0.57
StdFed	Model accuracy	0.86	0.80	0.81
	Accuracy on airplane class	0.55	0	0

different fraction of malicious clients and relabeled the samples with '0' to '1'. Similarly, the attack is also implemented using the CIFAR dataset by removing all airplanes from the clients' training data and relabelling all images of an airplane as 'ship' in the malicious clients' datasets [9]. Note that since only malicious clients have samples from the backdoor class, the detection of this attack is quite challenging.

Tables 10 and 11 display the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class to the targeted class (attack accuracy) for MNIST and CIFAR, respectively (values are chosen based on the best model accuracy over 100 rounds with MNIST and 300 rounds with CIFAR). We consider different fraction of malicious nodes. The results show that the model accuracy is similar for both datasets and schemes, but SignFed is much more robust against the attacks than StdFed. In fact, with 10% of malicious nodes, the attack accuracy on the MNIST dataset is very low for SignFed (19%) whereas it is quite large for StdFed (92%). We obtained similar pattern with the CIFAR dataset although the accuracy difference is less significant (66% versus 72%). This can be explained by the inherent bias present in CIFAR. For example, planes are often misclassified as 'bird' or 'ship' even without the attack because of the similar background of these images (i.e., sky is very similar to sea in many images). Indeed, the probability of predicting an airplane as a ship without the attack is

[9] we also removed all birds and trucks from the training data, in order to limit the bias between classes.

58%, and it only increases to 66% and 72% with SignFed and StdFed, respectively.

As for in-backdoor attacks, SignFed mitigates out-backdoor attacks because the adversary cannot scale up its update in order to increase its impact on the global model.

Table 10: Out-backdoors attack on SignFed and StdFed with the MNIST dataset. η_{Adv} is set to 1 (no boosting). The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "0" to the targeted class "1" (attack accuracy).

		10%	20%	40%
SignFed	Model accuracy	0.99	0.99	0.99
	Attack accuracy	0.19	0.87	0.99
StdFed	Model accuracy	0.99	0.99	0.99
	Attack accuracy	0.92	0.99	0.99

Table 11: Out-backdoors attack on SignFed and StdFed with the CIFAR dataset. η_{Adv} is set to 1 (no boosting). The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "airplane" to the targeted class "ship" (attack accuracy).

		10%	20%	40%
SignFed	Model accuracy	0.91	0.91	0.92
	Attack accuracy	0.66	0.74	0.93
StdFed	Model accuracy	0.92	0.92	0.90
	Attack accuracy	0.72	0.86	0.95

5.3 DP-SignFed Security Analysis

Table 12 and 13 depict the accuracy of the in-backdoor class and the misclassification rate of the out-backdoor class (best values are chosen based on the global model accuracy over 200 rounds) when backdoor attacks are launched against our DP schemes.

Malicious clients, whose fraction changes between 0.1 and 0.4, omit to add noise to their own updates at each round. We use the configuration described in Table 17 except for γ which is decreased to 0.001, and δ is fixed to 10^{-5}. In Fashion-MNIST dataset, at all malicious nodes, all images of 'Sandal' are relabelled to 'Sneaker' for In-backdoor, and all images of 'T-shirt/top' are relabelled to 'Trouser' for Out-backdoor attacks (only the malicious nodes have photos of 'T-shirt/top'). In addition, with DP-SignFed, each malicious node calculates their updates, extracts the signs ($\text{sign} : \mathbb{R}^n \rightarrow \{-1, 1\}^n$) and then uses a boosting parameter $\eta_{\text{Adv}} = 5000$ to boost their updates before sending them back to the server for aggregation. Indeed, as all honest clients send the noisy update in DP-SignFed, the noise together with encryption can conceal the manipulation of the malicious update vectors.

The results show that DP-SignFed are less robust against backdoor attacks than SignFed. On the MNIST dataset, model accuracy on the in-backdoor class is 0% for DP-SignFed regardless of the number of malicious nodes, and larger than 97% and 95% for SignFed, with 10% and 20% of malicious nodes, respectively. The same tendency holds for Fashion-MNIST. Out-backdoor attacks are especially effective on MNIST (see Table 14 and Table 15); here, the misclassification rate is more than 98% for DP-SignFed and 0-99% for SignFed. When we consider only 2% of malicious nodes with MNIST, the misclassification rate is 0% for SignFed and 76% for StdFed (without boosting) with a global model accuracy of 98% for both schemes. Indeed, StdFed is vulnerable to the outbackdoor attack even if we have only a small number malicious node and without using any boosting. For MNIST and Fashion-MNIST, SignFed is clearly superior to DP-SignFed regarding all attacks.

Finally, random update attack and gradient ascent attack are mounted against DP-SignFed. The same parameters are used as in the previous experiments. Malicious clients still omit to add any noise to their own model updates. Instead, they boost their signs updates with DP-SignFed ($\eta_{\text{Adv}} = 5000$). The model fails to converge even if only 1% of all selected nodes are malicious at each round.

Table 12: In-backdoor attack on DP-SignFed and SignFed with MNIST dataset. The table depicts the global accuracy convergence and the accuracy of the label "5" which is under attack.

		10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.89	0.90	0.90
	Accuracy on digit class '5'	0	0	0
$\varepsilon = 2$	Model accuracy	0.89	0.90	0.90
	Accuracy on digit class '5'	0	0	0
$\varepsilon = 4$	Model accuracy	0.89	0.90	0.90
	Accuracy on digit class '5'	0	0	0
SignFed	Model accuracy	0.98	0.98	0.90
	Accuracy on digit class '5'	0.97	0.95	0

Table 13: In-backdoor attack on SignFed and DP-SignFed with Fashion-MNIST dataset. The table depicts the global accuracy convergence and the accuracy of the label "Sandal" which is under attack.

		10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.77	0.79	0.80
	Accuracy on Sandal class	0	0	0
$\varepsilon = 2$	Model accuracy	0.77	0.79	0.80
	Accuracy on Sandal class	0	0	0
$\varepsilon = 4$	Model accuracy	0.77	0.79	0.80
	Accuracy on Sandal class	0	0	0
SignFed	Model accuracy	0.83	0.84	0.79
	Accuracy on Sandal class	0.90	0.84	0

6 RELATED WORK

Security of Federated Learning: Federated learning being a relatively new concept, its security has not been studied so far. However, most ML security attacks also apply to it. In this section, we mostly

Table 14: Out-backdoors attack on DP-SignFed and SignFed with MNIST dataset. The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "0" to the targeted class "1" (attack accuracy).

		2%	10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.98	0.98	0.99	0.99
	Attack accuracy	0.98	0.99	0.99	1
$\varepsilon = 2$	Model accuracy	0.98	0.98	0.98	0.99
	Attack accuracy	0.99	0.98	0.99	0.99
$\varepsilon = 4$	Model accuracy	0.98	0.98	0.99	0.99
	Attack accuracy	0.99	0.99	0.99	0.99
SignFed	Model accuracy	0.98	0.98	0.98	0.99
	Attack accuracy	0	0.97	0.99	0.99

Table 15: Out-backdoors attack on DP-SignFed and FL-SIG with Fashion-MNIST dataset. The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "T-shirt/Top" to the targeted class "Trouser" (attack accuracy).

		10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.87	0.88	0.90
	Attack accuracy	0.78	0.81	0.87
$\varepsilon = 2$	Model accuracy	0.88	0.88	0.90
	Attack accuracy	0.78	0.82	0.85
$\varepsilon = 4$	Model accuracy	0.87	0.89	0.90
	Attack accuracy	0.78	0.81	0.86
SignFed	Model accuracy	0.87	0.88	0.90
	Attack accuracy	0	0.12	0.83

focus on integrity attacks [33]. These attacks include *pollution* and *backdoor attacks*.

In pollution attacks, the adversary manipulates its training data or the corresponding labels to poison the global model. In backdoor attacks, the adversary manipulates its training data or the incoming labels in order to insert backdoors into the global model. Indeed, the goal of the adversary is to cause the misclassification of specific labels in the global model while maintaining a good accuracy on the non-targeted labels. In the context of Federated Learning, the most efficient strategy consists of directly manipulating the model updates.

The first backdoor attack designed for a federated learning environment was proposed in [6]. Here, the adversary scales up its update in order to surpass the contributions of other honest participants after aggregation. The goal of the attack is to alter the common model so that it exhibits some adversarial behaviour (e.g., targeted misclassification). However, these attacks are effective only in later rounds, when the global model has converged. Indeed, the attack exploits the fact that when the global model has converged, the updates of other honest clients will be smaller and then are more easier to surpass. In contrast, the adversary in [9] boosts its update enough to surpass the contributions of the honest clients from the very first rounds even when the global model has not converged.

The resilience of distributed implementations of Stochastic Gradient Descent (SGD) against Byzantine failures is studied in [10]. Each Byzantine worker (among a set of workers) sends a random vector drawn from a Gaussian distribution. The results show that only a single Byzantine worker can prevent the traditional federated schemes such as StdFed from converging. In the same paper, KRUM a Byzantine-resilient algorithm is proposed as an aggregation rule to select one honest update per round in an adversarial environment.

In [8], the authors study the robustness and the tolerance of signSGD/SIGNUM [7] with majority vote against network faults and adversarial clients, where SIGNUM is the momentum equivalent of signSGD (i.e., each client maintains a momentum and transmits the sign momentum to the server at each iteration). In [8], the authors show that signSGD is robust against sign inversion attack, when each malicious client inverts the sign of the computed gradient. The authors argue that this is the best possible attack in a *non-adaptive* setting (i.e., when the adversary performs the attack independently of the gradients it computed). In this paper, we experimentally show that SignFed is also robust against other adaptive attacks like various backdoor attacks [9].

Privacy of Federated Learning: There exist a few inference attacks specifically designed against federated learning schemes. In [28], the adversary's goal is to infer whether records with a specific property are included in the training dataset of the other participants (called batch property inference). The authors demonstrate the attack by inferring whether black people are included in any of the training datasets, where the common model is trained for gender classification (i.e., the inferred property is independent of the learning objective). The adversary is supposed to have access to the aggregated model update of honest participants. In [31], the proposed attack infers if a specific person is included in the training dataset of the participants (aka, Membership inference). The adversary extracts the following features from every snapshot of the common model, which is a neural network: output value, hidden layers, loss values, and the gradient of the loss with respect to the parameters of each layer. These features are used to train a membership inference model, which is a convolutional neural network.

The concept of Client-based Differential Privacy has been introduced in [27] and [18], where the goal is to hide any information that is specific to a single client's training data. These algorithms bound and noise the contribution of a single client's instead of a single record in the client's dataset. The noise is added by the server, hence, unlike our solution, these works assume that the server is trusted. Also, the noise is drawn from continuous distributions.

Bandwidth Optimization in Federated Learning: Different quantization methods have been proposed to save the bandwidth and reduce the communication costs in federated learning. They can be divided into two main groups: unbiased and biased methods. The unbiased approximation techniques use probabilistic quantization schemes to compress the stochastic gradient and attempt to approximate the true gradient value as much as possible [5][40][39][21]. However, biased approximations of the stochastic gradient can still

guarantee convergence both in theory and practice [7, 25, 34]. In signSGD [7], all the clients calculate the stochastic gradient based on a single mini-batch and then send the sign vector of this gradient to the server. The server calculates the aggregated sign vector by taking the median (majority vote) and sends the signs of the aggregated signs back to each client.

The main differences between our scheme (SignFed) and signSGD are as follows:

- SignFed aims to train a common model that is distributed to a random subset of all clients in every round. However, in signSGD, each client builds its own model locally and the server sends the same aggregated model update to *every* client. Selecting only a random subset of clients in each round has at least three benefits. First, SignFed becomes more robust against temporary node failures. Second, SignFed reduces the communication costs upstream to the server. Finally, sampling boosts privacy due to the uncertainty that a specific user's or client's data is used for training or not.
- In SignFed, each client can perform multiple SGD iterations locally using multiple mini-batches before computing the model update. On the contrary, signSGD always performs one local SGD iteration with a single mini-batch at every client. This is needed to guarantee convergence since all nodes maintain different local models unlike in SignFed.
- As there is no single common model built in signSGD, the server only transfers the sign of the aggregated signs to the clients in every round. Therefore, only a single bit is transferred per parameter downstream to the clients. In SignFed, the whole model is transferred but only to a random subset of clients.

7 SUMMARY AND DISCUSSION

We can make the following main observations.

(1) SignFed is almost as accurate as StdFed but incurs less communication overhead and has better resiliency against both security and privacy attacks (see Section 3.3 and 5.2).
(2) Although SignFed is more robust against state-of-the-art privacy attacks than StdFed, DP-SignFed provides provable privacy guarantees unlike SignFed. However, it also produces models with slightly worse accuracy than SignFed. More importantly, it is less robust against security attacks (see Section 4.3 and 5.3).
(3) DP-SignFed has 30-40% less communication cost than StdFed but it is roughly 20 times more than that of SignFed. In DP-SignFed, there is a trade-off between privacy and bandwidth. Stronger privacy requires to increase the variance of the added discrete Gaussian noise which in turn implies larger communication costs (see Section 4.2).
(4) The convergence rates of SignFed and DP-SignFed are $O\left(\frac{1}{\sqrt{T_{\text{cl}}CN}}\right)$ and $O\left(\frac{1}{\sqrt{T_{\text{cl}}CN}} + \frac{n^{3/2}\sigma}{\sqrt{T_{\text{cl}}CN}}\right)$, respectively, supposing that $\gamma = O(1/\sqrt{T_{\text{cl}}})$, $T_{\text{gd}} = 1$, $|\mathbb{B}| = T_{\text{cl}}$ (see Appendix A.4 for the proofs). Therefore, the "cost of privacy" in convergence rate is $O\left(\frac{n^{3/2}\sigma}{\sqrt{T_{\text{cl}}CN}}\right)$ which is due to the added noise.

Seemingly, there is a possible trade-off between differential privacy and robustness against security attacks. One possible explanation is that differential privacy requires to randomize every value of the update vector so much that their aggregates become easier to manipulate. As malicious clients omit to add any noise to their own model updates, the attacked DP protocols essentially turn into Random Update Attacks, where honest clients send almost uniformly random signs and malicious clients transfer non-noisy, boosted updates to the aggregator. As SignFed converges with Random Update Attack even with limited number of honest nodes, the malicious nodes in DP-SignFed can also degrade model performance or inject backdoors for the very same reason. The smaller ε is the more uniform every coordinate's distribution will be, and the larger impact a malicious client has on the aggregate.

REFERENCES

[1] Martín Abadi, , et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/ Software available from tensorflow.org.
[2] M. Abadi, A. Chu, I. Goodfellow, H. Brendan McMahan, I. Mironov, K. Talwar, and L. Zhang. 2016. Deep Learning with Differential Privacy. In *ACM CCS* (Vienna, Austria). ACM, New York, NY, USA, 308–318.
[3] Anonymized according to the Submission guidelines. 2020. Technical report. (2020).
[4] G. Ács and C. Castelluccia. 2011. I Have a DREAM! (DiffeRentially privatE smArt Metering). In *Information Hiding - 13th International Conference, IH 2011*. 118–132.
[5] D. Alistarh, J. Li, R. Tomioka, and M. Vojnovic. 2016. QSGD: Randomized Quantization for Communication-Optimal Stochastic Gradient Descent. *CoRR* abs/1610.02132 (2016). arXiv:1610.02132 http://arxiv.org/abs/1610.02132
[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2018. How To Backdoor Federated Learning. *CoRR* abs/1807.00459 (2018). arXiv:1807.00459 http://arxiv.org/abs/1807.00459
[7] J. Bernstein, Y. Wang, K. Azizzadenesheli, and A. Anandkumar. 2018. signSGD: compressed optimisation for non-convex problems. *CoRR* abs/1802.04434 (2018). arXiv:1802.04434 http://arxiv.org/abs/1802.04434
[8] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar. 2018. signSGD with Majority Vote is Communication Efficient And Byzantine Fault Tolerant. *CoRR* abs/1810.05291 (2018). arXiv:1810.05291 http://arxiv.org/abs/1810.05291
[9] A. Nitin Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo. 2018. Analyzing Federated Learning through an Adversarial Lens. *CoRR* abs/1811.12470 (2018).
[10] P. Blanchard, El M. El Mhamdi, R. Guerraoui, and J. Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NIPS*. 119–129.
[11] K. Bonawitz et al. 2016. Practical Secure Aggregation for Federated Learning on User-Held Data. (2016).
[12] F. Chollet et al. 2015. Keras. https://keras.io.
[13] F. Chollet et al. 2015. Keras datasets. https://keras.io/datasets/.
[14] F. Chollet et al. 2015. Keras optimizers. https://keras.io/optimizers/.
[15] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das. 2019. Differential Privacy-enabled Federated Learning for Sensitive Health Data. arXiv:1910.02578 [cs.LG]
[16] C. Dwork and A. Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014).
[17] Ú. Erlingsson, V. Pihur, and A. Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *ACM SIGSAC Conference on Computer and Communications Security, 2014*. 1054–1067.
[18] R. C. Geyer, T. Klein, and M. Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. *CoRR* abs/1712.07557 (2017). arXiv:1712.07557
[19] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts, Amherst.
[20] D. P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. (2014).
[21] J. Konecný, H. Brendan McMahan, F. X. Yu, Peter Richtárik, A. T. Suresh, and D. Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR* abs/1610.05492 (2016). arXiv:1610.05492
[22] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[23] Mate Labs. 2017. *How these researchers tried something unconventional to come out with a smaller yet better Image Recognition.* Retrieved July 29, 2019 from https://medium.com/@matelabs_ai/how-these-researchers-tried-something-unconventional-to-came-out-with-a-smaller-yet-better-image-544327f30e72#.i3227lyhb
[24] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[25] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations, ICLR 2018.*
[26] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS.*
[27] H. Brendan McMahan, D. Ramage, K. Talwar, and L. Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations.* https://openreview.net/forum?id=BJ0hF1Z0b
[28] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. 2018. Inference Attacks Against Collaborative Learning. *CoRR* abs/1805.04049 (2018). arXiv:1805.04049
[29] D. Micciancio and M. Walter. 2017. Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time. In *Advances in Cryptology - CRYPTO 2017.* 455–485.
[30] I. Mironov, K. Talwar, and L. Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *CoRR* abs/1908.10530 (2019). arXiv:1908.10530
[31] M. Nasr, R. Shokri, and A. Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy, 2019.* 739–753.
[32] Travis E Oliphant. 2006. *A guide to NumPy.* Vol. 1. Trelgol Publishing USA.
[33] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman. 2018. SoK: Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy, EuroS&P, 2018.* 399–414.
[34] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *INTERSPEECH 2014.* 1058–1062.
[35] R. Shokri and V. Shmatikov. 2015. Privacy-Preserving Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security, 2015.* 1310–1321.
[36] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. 2015. Striving for Simplicity: The All Convolutional Net. In *ICLR (workshop track).* http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a
[37] Paweł J. Szabłowski. 2001. Discrete Normal distribution and its relationship with Jacobi Theta functions. *Statistics & Probability Letters* 52, 3 (2001), 289 – 299.
[38] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, and R. Zhang. 2018. A Hybrid Approach to Privacy-Preserving Federated Learning. (2018).
[39] H. Wang, S. Sievert, S. Liu, Z. B. Charles, D. S. Papailiopoulos, and S. Wright. 2018. ATOMO: Communication-efficient Learning via Atomic Sparsification. In *NeurIPS.*
[40] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. 2017. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. (2017).
[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR* abs/1708.07747 (2017). arXiv:1708.07747 http://arxiv.org/abs/1708.07747

A APPENDIX

A.1 Proof of Lemma 1

Proof. We first show that $Z/\sqrt{2\pi}\xi \le 1 + \frac{2e^{-2\pi^2\xi^2}}{1-e^{-6\pi^2\xi^2}}$, where $Z = \sum_{x\in\mathbb{Z}} \exp(-(x-\mu)^2/2\xi^2)$, which implies the upper bound. From [37], $Z = \sqrt{2\pi}\xi\vartheta_3(\pi\mu, \exp(-2\pi^2\xi^2))$, where $\vartheta_3(u,r) = 1 + 2\sum_{i\ge1} r^{i^2}\cos(2iu)$ is a Jacobi Theta function. Then,

$$1 + 2\sum_{i\ge1} r^{i^2}\cos(2iu) \le 1 + 2r\sum_{i\ge0} r^{3i} \le 1 + \frac{2r}{1-r^3}$$

if $|r| < 1$. The lower bound can be derived similarly using the fact that $\cos(2iu) \ge -1$. □

A.2 Proof of Theorem 1

Proof. Without loss of generality, suppose that $\mathcal{DG}_\xi : \mathbb{R} \to \mathbb{Z}^n$.

We apply the moments accountant [2] and show that $\alpha_{\mathcal{DG}}(\lambda)$ can be upper bounded efficiently without evaluating the pmf of $\mathcal{DG}$.

Let $\eta_0^{\mathcal{DG}}(x|\xi) = \mathrm{pmf}_{\mathcal{DG}(0,\xi)}(x)$ and $\eta_1^{\mathcal{DG}}(x|\xi) = (1-q)\mathrm{pmf}_{\mathcal{DG}(0,\xi)}(x) + q\mathrm{pmf}_{\mathcal{DG}(1,\xi)}(x)$ where $\mathrm{pmf}_{\mathcal{DG}(\mu,\xi)}(x) = \Pr_{x\sim\mathcal{DG}(\mu,\xi)}[x]$. Then,

$$\alpha_{\mathcal{DG}}(\lambda) = \log\max(E_1'(\lambda,\xi), E_2'(\lambda,\xi))$$

where

$$\begin{aligned} E_1'(\lambda,\xi) &= \sum_{x=-\infty}^{\infty} \eta_0^{\mathcal{DG}}(x|\xi)\cdot\left(\frac{\eta_0^{\mathcal{DG}}(x|\xi)}{\eta_1^{\mathcal{DG}}(x|\xi)}\right)^\lambda \\ &\le \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}\sum_{x=-\infty}^{\infty}\eta_0^{\mathcal{G}}(x|\xi)\cdot\left(\frac{\eta_0^{\mathcal{G}}(x|\xi)}{\eta_1^{\mathcal{G}}(x|\xi)}\right)^\lambda \\ &\le \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}\int_{\mathbb{R}}\eta_0^{\mathcal{G}}(y|\xi)\cdot\left(\frac{\eta_0^{\mathcal{G}}(y|\xi)}{\eta_1^{\mathcal{G}}(y|\xi)}\right)^\lambda dy \\ &= \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}E_1(\lambda,\xi) \end{aligned}$$

where the first inequality follows from Lemma 1. Using a similar reasoning we obtain that

$$E_2'(\lambda,\xi) \le \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}E_2(\lambda,\xi)$$

The theorem follows from Theorem 2 in [2]. □

A.3 Proof of Theorem 2

As opposed to the continuous case, the sum of discrete Gaussian random variables $\sum_i \mathcal{DG}(\mu_i,\xi_i)$ does not follow the distribution of $\mathcal{DG}(\sum_i\mu_i, \sqrt{\sum_i\xi_i^2})$, though it is very close to that if ξ_i is sufficiently large. The exact difference is quantified by the following Lemma from [29].

Lemma 3 ([29], Theorem 2.1). *If $\xi_i \ge \sqrt{\ln(2+2/\nu)}/\pi$ and $\boldsymbol{\mu}_i \in \mathbb{R}^n$, then*

$$\frac{1-\nu}{1+\nu} \le \frac{\Pr_{\mathbf{z}\sim\sum_i\mathcal{DG}(\boldsymbol{\mu}_i,\xi_i)}[\mathbf{z}]}{\Pr_{\mathbf{z}\sim\mathcal{DG}\left(\sum_i\boldsymbol{\mu}_i,\sqrt{\sum_i\xi_i^2}\right)}[\mathbf{z}]} \le \frac{1+\nu}{1-\nu}$$

for any $\mathbf{z}\in\mathbb{Z}^n$

Intuitively, if $Z = \sum_{x\in\mathbb{Z}}\exp(-(x-\mu)^2/2\xi_i^2) \approx \sqrt{2\pi}\xi_i$ then $\mathcal{DG}(\mu_i,\xi_i) \approx \mathcal{G}(\mu_i,\xi_i)$, in which case $\sum_i\mathcal{DG}(\mu_i,\xi_i) \approx \sum_i\mathcal{G}(\mu_i,\xi_i) = \mathcal{G}(\mu_i,\sum_i\xi_i) \approx \mathcal{DG}(\mu_i,\sum_i\xi_i)$, which follows from Lemma 2. Indeed, it also follows from the proof of Lemma 1 that $Z/\sqrt{2\pi}\xi_i \le \vartheta_3(\pi\mu,\exp(-2\pi^2\xi_i^2)) \le 1 + \frac{2e^{-2\pi^2\xi_i^2}}{1-e^{-6\pi^2\xi_i^2}} \le 1 + \frac{2}{\exp(2\xi_i^2\pi^2)-1} \le 1 + \frac{2}{\exp(\xi_i^2\pi^2)-2} \le 1+\nu$ which provides some insight into the condition on ξ_i in Lemma 3.

For example, $\nu < 10^{-4}$ is satisfied if $\xi_i > 1$.

Let $\widehat{\mathcal{DG}}_\xi$ denote the distributed Gaussian mechanism which returns $\sum_{k=1}^N\mathcal{DG}(\boldsymbol{\mu}_k,\xi/\sqrt{|\mathbb{K}|})$ where $\boldsymbol{\mu}_k\in\mathbb{R}^n$. The next lemma, which directly follows from Theorem 1 and Lemma 3, implies Theorem 2.

Lemma 4. *If $\xi \ge \sqrt{|\mathbb{K}|\ln(2+2/\nu)}/\pi$, then $\alpha_{\widehat{\mathcal{DG}}}(\lambda|q) \le \alpha_{\mathcal{G}}(\lambda|q) + \log\left(\frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}\left(\frac{1+\nu}{1-\nu}\right)^3\right)$.*

A.4 Convergence Proofs

All the proofs are simple adaptations of Theorem 2 from [8]. Here we outline only the main deviations from the proof of that theorem.

Assumptions:

(1) *Lower bound:* For all x and some constant f^*, $f(x) \geq f^*$, where f denotes the loss/objective function.
(2) *Smoothness:* Let $g(x)$ denote the gradient of the objective function f evaluated at x. Then, for all x, y and some non-negative constant $\mathbf{L} = (L_1, L_2, \ldots, L_n)$,

$$|f(y) - [f(x) + g(x)^\mathsf{T}(y - x)]| \leq 1/2 \sum_i L_i (y_i - x_i)^2$$

(3) *Variance bound:* Upon receiving query $x \in \mathbb{R}^n$, the stochastic gradient oracle gives us an independent, unbiased estimate $\hat{g}$ that has bounded variance per coordinate: $\mathbb{E}[\hat{g}(x)] = g(x)$, $\mathbb{E}[(\hat{g}(x)_i - g(x)_i)^2] \leq \tau_i^2$ for a vector of non-negative constants $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_n)$.
(4) *Unimodal, symmetric gradient noise:* At any given point x, each component of the stochastic gradient vector $\hat{g}(x)$ has unimodal distribution that is also symmetric about the mean.

Note that adding extra Gaussian noise to each gradient component for the purpose of differential privacy will not violate Assumption 4.

Theorem 3. *If* $|\mathbb{B}| = T_{\text{cl}}$, $T_{\text{gd}} = 1$, *and* $\gamma = \sqrt{\frac{f_0 - f_*}{||\mathbf{L}||_1 T_{\text{cl}}}}$, *then*

(1) For SignFed in the Random Update Attack,

$$\frac{1}{T_{\text{cl}}} \sum_{t=0}^{T_{\text{cl}}-1} \mathbb{E}||g_t||_1 \leq \frac{2}{\sqrt{T_{\text{cl}}}} \left(\frac{\sqrt{2}||\boldsymbol{\tau}||_1}{(1-\alpha)\sqrt{CN}} + \sqrt{||\mathbf{L}||_1 (f_0 - f^*)} \right)$$

where α denotes the fraction malicious clients and $|g_i|/\tau_i \leq 2/\sqrt{3}$ for all $1 \leq i \leq n$.
(2) For DP-SignFed,

$$\frac{1}{T_{\text{cl}}} \sum_{t=0}^{T_{\text{cl}}-1} \mathbb{E}||g_t||_1 \leq \frac{2}{\sqrt{T_{\text{cl}}}} \left(\frac{||\boldsymbol{\tau}||_1}{\sqrt{CN}} + \frac{\sqrt{3n}\sigma||\boldsymbol{\tau}||_1}{CN} + \sqrt{||\mathbf{L}||_1 (f_0 - f^*)} \right)$$

if $|g_i|/\tau_i \leq 2/\sqrt{3}$ for all $1 \leq i \leq n$.

Proof. The primary focus of all the proofs is to bound the probability that a client computes the sign of a parameter update correctly. Let $M = CN$. As in [8], let $Z_i \in [0, M]$ denote the number of correct sign bits received by the aggregator for parameter i, and p denotes the probability that a honest client computes the correct bit. Let $\omega = p - \frac{1}{2}$.

(1) **Random Update Attack:** Notice that the probability that a sign of any parameter is correct at a malicious client is 1/2, and each client acts independently from each other. Hence, $\mathrm{E}[Z_i] = (1-\alpha)Mp + \frac{1}{2}\alpha M$ and $\mathrm{Var}[Z] = \frac{1}{4}\alpha M + (1-\alpha)Mp(1-p)$. The probability that a vote fails for the i^{th} parameter is identical to $\mathrm{P}[Z_i \leq M/2]$, which, likewise in [8], can be bounded as follows.

$$\begin{aligned}
\mathrm{P}[Z_i \leq M/2] &= \mathrm{P}[\mathrm{E}[Z_i] - Z_i \geq \mathrm{E}[Z_i] - M/2] \\
&\leq \frac{1}{1 + \frac{(\mathrm{E}[Z_i] - N/2)^2}{\mathrm{Var}[Z_i]}} \qquad \text{(by Cantelli's inequality)} \\
&\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i]}{(\mathrm{E}[Z_i] - M/2)^2}} \qquad \text{(by } 1 + x^2 \geq 2x\text{)} \\
&\leq \frac{1}{2\sqrt{M}}\sqrt{\frac{\frac{1}{4}\alpha + (1-\alpha)p(1-p)}{(1-\alpha)^2(p-\frac{1}{2})^2}} \\
&\leq \frac{1}{2\sqrt{M}}\sqrt{\frac{\frac{1}{4}\alpha}{(1-\alpha)^2(p-\frac{1}{2})^2}} \\
&\quad + \frac{1}{2\sqrt{M}}\sqrt{\frac{p(1-p)}{(1-\alpha)(p-\frac{1}{2})^2}} \\
&\leq \frac{\sqrt{\alpha}}{4\sqrt{M}(1-\alpha)|\omega|} + \frac{1}{2\sqrt{M}}\sqrt{\frac{\frac{1}{4} - \omega^2}{(1-\alpha)\omega^2}} \\
&\leq \frac{\sqrt{3\alpha}\tau_i}{2\sqrt{M}(1-\alpha)|g_i|} + \frac{\tau_i}{\sqrt{M(1-\alpha)}|g_i|} \qquad (5) \\
&\leq \frac{\tau_i(\sqrt{\alpha} + \sqrt{1-\alpha})}{\sqrt{M}(1-\alpha)|g_i|} \\
&\leq \frac{\sqrt{2}\tau_i}{\sqrt{M}(1-\alpha)|g_i|}
\end{aligned}$$

where, in Eq. (5), we used that $\frac{1}{4\omega^2} - 1 \leq 4\tau_i^2/g_i^2$ and $1/|\omega| \leq 2\sqrt{3}\tau_i/|g_i|$ for $|g_i|/\tau_i < 2/\sqrt{3}$ based on Lemma 1 in [8]. The rest of the derivation is identical to the proof of Theorem 2 in [8].

(2) **DP-SignFed:** The Gaussian noise is added to the sum of signs. Let Y_i denote the random variable describing the noise added by the clients to Z_i.

$$\begin{aligned}
\mathrm{P}[Z_i + Y_i \leq M/2] &\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i + Y_i]}{(\mathrm{E}[Z_i + Y_i] - M/2)^2}} \\
&\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i] + \mathrm{Var}[Y_i]}{(\mathrm{E}[Z_i] - M/2)^2}} \\
&\qquad \text{(by independence and } \mathrm{E}[Y_i] = 0\text{)} \\
&\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i]}{(\mathrm{E}[Z_i] - M/2)^2}} \\
&\quad + \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Y_i]}{(\mathrm{E}[Z_i] - M/2)^2}} \qquad (6)
\end{aligned}$$

Based on [8],

$$\frac{1}{2}\sqrt{\frac{\text{Var}\left[Z_i\right]}{(\text{E}\left[Z_i\right]-M/2)^2}} \leq \frac{1}{2}\sqrt{\frac{M\left(\frac{1}{4}-\omega^2\right)}{M^2\omega^2}} \leq \frac{1}{2}\sqrt{\frac{M4\tau_i/|g_i|}{M^2\omega^2}} \leq \frac{\tau_i}{\sqrt{M}|g_i|} \quad (7)$$

Moreover, if $|g_i|/\tau_i \leq 2/\sqrt{3}$, then $1/\omega^2 \leq 12\tau_i^2/g_i^2$, and hence

$$\frac{1}{2}\sqrt{\frac{\text{Var}\left[Y_i\right]}{(\text{E}\left[Z_i\right]-M/2)^2}} \leq \frac{1}{2}\sqrt{\frac{n\sigma^2}{M^2\omega^2}} \leq \frac{1}{2}\sqrt{\frac{12n\sigma^2\tau_i^2/g_i^2}{M^2}} \leq \frac{\sqrt{3}\sqrt{n}\sigma\tau_i}{M|g_i|} \quad (8)$$

Plugging Eq. (7) and (8) into Eq. (6), we obtain that the probability that the noisy vote fails for the i^{th} coordinate is bounded as

$$\text{P}\left[Z_i + Y_i \leq M/2\right] \leq \frac{\tau_i}{\sqrt{M}|g_i|} + \frac{\sqrt{3}\sqrt{n}\sigma\tau_i}{M|g_i|}$$

if $|g_i|/\tau_i \leq 2/\sqrt{3}$. The claim follows from the proof of Theorem 2 in [8].

□

A.5 Model architectures

For MNIST and Fashion-MNIST, we use a model [26] with the following architecture: a convolutional neural network (CNN) with two 5x5 convolution layers (the first with 32 filters, the second with 64, each followed with 2x2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer. This results in 1,663,370 parameters in total.

The LFW dataset is used with a CNN of three 3x3 convolution layers (32, 64, and 128 filters, each followed with 2x2 max pooling), a fully connected layer with 256 units and ReLU activation, and a final softmax output layer with 2 units. To test the property inference attack from [28], batch size is set to 32, and the SGD learning rate is 0.01.

The model that we use for the CIFAR dataset is called "All-CNN-C" in [36] [23], which consists of a CNN of 3 blocks: the first block has three 3x3 convolutions layers with 96 filters (the last layer has a strides of 2x2 and dropout of 0.5 is applied), the ReLu activation is used per layer. The second block has the same configuration as the previous block, except the filter size which is 192 for each layer. The last block has one 3x3 convolutions layer with 192 filters, followed by two 1x1 convolution layers: the first with 192 filters (Relu activation) and the second with 10 filters. The last layer is connected with a global average pooling layer and uses softmax activation. We use also the Adam optimizer with a learning rate of 0.001. This results in 1,369,738 parameters in total.

Finally, we use the following model for the IMDB dataset: one embedding layer with an output size of 50 (the vocabulary size is set to 5000 and the maximum length input to 400), followed by a convolution layer of one dimension with a kernel size of 5 and 250 filters; and a max pooling layer of size 3; followed by a LSTM layer with an output size set to 70 and an output layer with one unit that uses a sigmoid activation function. We use the Adam optimizer with a learning rate of 0.001. This results in 402,701 parameters in total.

A.6 Selection of hyperparameters

Strictly speaking, the selection of hyperparameters in DP-SignFed, such as batch size $|\mathbb{B}|$, scaling factor γ, or sensitivity S, must also be differentially private. One option is to use public data for this purpose which comes from the same distribution as the clients' private training data. The selection of hyperparameters can also be performed using more sophisticated methods like the one in Appendix D of [2].

A.7 Robustness of DP-SignFed against client failures

If any client fails to add its noise share to the model update for any reason, the aggregate will not have sufficient amount of noise to guarantee differential privacy. A straightforward countermeasure is to increase the variance of the added noise so that even if r clients fail, the sum of $CN - r$ noise shares are still enough for differential privacy. In particular, each client k sends $\text{Enc}_{K_k}(\mathcal{DG}(\text{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{CN-r}))$ to the server for aggregation. Obviously, if less than r nodes fail, the aggregate will have larger noise than what is necessary for differential privacy.

Table 16: Common environment and configuration of SignFed and StdFed. $\gamma = 0.001$.

Datasets	MNIST Fashion-MNIST	IMDB	CIFAR
Parameters	$N = 1000$; $C = 0.1$; $\|D_k\| = 60$; $\|\mathbb{B}\| = 10$; $T_{\text{gd}} = 30$; $T_{\text{cl}} = 100$; *SGD* ($\eta = 0.215$)	$N = 1000$; $C = 0.1$; $\|D_k\| = 25$; $\|\mathbb{B}\| = 25$; $T_{\text{gd}} = 5$; $T_{\text{cl}} = 100$; *ADAM* ($\eta = 0.001$)	$N = 1000$; $C = 0.1$; $\|D_k\| = 500$; $\|\mathbb{B}\| = 50$; $T_{\text{gd}} = 50$; $T_{\text{cl}} = 400$; *ADAM* ($\eta = 0.001$)

A.8 Computational Environment

Our experiments were performed on a server running Ubuntu 18.04 LTS equipped with a Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 192GB RAM, and two NVIDIA Quadro P5000 GPU card of 16 Go each. We use Keras 2.2.0 [12] with a TensorFlow backend 1.12.0 [1] and Numpy 1.14.3 [32] to implement our models and experiments. We use Python 3.6.5 and our code runs on a Docker container to simplify the reproducibility.

A.9 Datasets

The following datasets were used:

Table 17: Common environment of the privacy part. $\gamma = 0.005$ and $T_{\text{cl}} = 200$. For DP-StdFed, S is set to 1.73 and 2.15 for MNIST and Fashion-MNIST, respectively. For DP-SignFed, S is fixed to $\sqrt{n}$.

Algorithms \ Datasets	MNIST & Fashion-MNIST
DP-SignFed & DP-StdFed	$N = 6000$; $C = 1/60$; $\lvert D_k\rvert = 10$; $\lvert\mathbb{B}\rvert = 10$; $T_{\text{gd}} = 5$; $SGD(\eta = 0.215)$; $n = 1,663,370$; $\delta = 10^{-5}$

Table 18: Parameter Configuration for the Backdoor Attacks. SignFed is used with the vote aggregation $\gamma = 0.001$.

Attacks \ Datasets	MNIST
In-backdoor	$N = 10$; $C = 1$; $\lvert D_k\rvert = 6000$; $\lvert\mathbb{B}\rvert = 10$; $T_{\text{gd}} = 3000$; $T_{\text{cl}} = 40$; $SGD(\eta = 0.215)$
Out-backdoor	$N = 10$; $C = 1$; $\lvert D_k\rvert = 6000$; $\lvert\mathbb{B}\rvert = 10$; $T_{\text{gd}} = 3000$; $T_{\text{cl}} = 100$; $SGD(\eta = 0.215)$
Attacks \ Datasets	**CIFAR**
In-backdoor	$N = 10$; $C = 1$; $\lvert D_k\rvert = 50000$; $\lvert\mathbb{B}\rvert = 100$; $T_{\text{gd}} = 1000$; $T_{\text{cl}} = 100$; $ADAM(\eta = 0.001)$
Out-backdoor	$N = 10$; $C = 1$; $\lvert D_k\rvert = 50000$; $\lvert\mathbb{B}\rvert = 100$; $T_{\text{gd}} = 1000$; $T_{\text{cl}} = 300$; $ADAM(\eta = 0.001)$

- The MNIST database of handwritten digits. It consists of 28 x 28 grayscale images of digit items and has 10 output classes. The training set contains 60,000 data samples while the test/validation set has 10,000 samples [24] [13].
- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We augment the dataset to 500,000 training images by randomly shifting the original images horizontally and vertically and by randomly flipping the original images horizontally [22] [13].
- Fashion-MNIST database of fashion articles consists of 60,000 28x28 grayscale images of 10 fashion categories, along with a test set of 10,000 images [41] [13].
- IMDB Movie reviews sentiment classification dataset of 25,000 movies reviews, labeled by sentiment (positive/negative) [13]. The test set contains also 25,000 movies reviews.

Algorithm 5: DP-StdFed: Federated Learning with Client Privacy

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients randomly
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta\tilde{\mathbf{w}}_t^k = \textbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \frac{1}{|\mathbb{K}|}\sum_k \Delta\tilde{\mathbf{w}}_t^k$
  end
Client$_k(\mathbf{w})$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}$
  $\Delta\mathbf{w}_t^k = \textbf{SGD}(D_k, \mathbf{w}_t^{k-1}, T_{\text{gd}}) - \mathbf{w}_{t-1}^k$
  $\Delta\hat{\mathbf{w}}_t^k = \Delta\mathbf{w}_t^k / \max\left(1, \frac{||\Delta\mathbf{w}_t^k||_2}{S}\right)$
  Output: $\text{Enc}_{K_k}(\mathcal{G}(\Delta\hat{\mathbf{w}}_t^k, S\text{I}\sigma/\sqrt{|K|}))$

- Labeled Faces in the Wild (LFW) dataset: consists of 13,000 $62 \cdot 47$ RGB images of faces collected from the web [19].